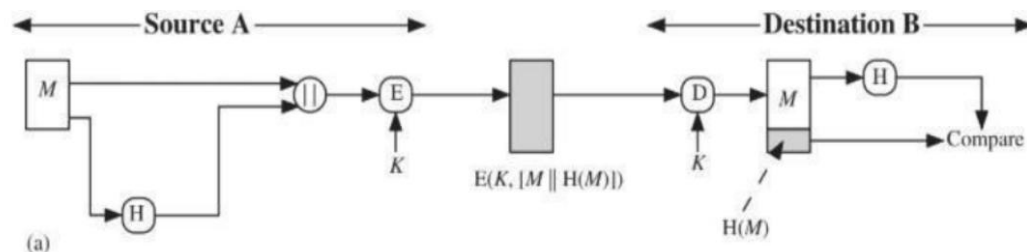


PRACTICAL: 6

AIM:

Refer to the figure (a) attached here. Bob (Source A) is preparing to send a message to Alice (Destination B). Bob applies the SHA256 hash algorithm on the prepared message and appends it with original message (M) which is further encrypted by a single secret key. Alice will receive a bundle of encrypted $H(M)$ and original messages (M). Alice will first apply a single secret key to decrypt the entire bundle and collect $H(M)$ and the original message (M). Furthermore, Alice will apply the same algorithm SHA256 which was used by Bob, and produce a hash of the received message (H). Lastly, Alice will verify the computed hash with the received $H(M)$ to make sure the message is not altered by any attackers.



Task to perform:

1. Use any Symmetric key/Asymmetric key algorithm to implement encryption function and decryption.
2. Implementation can be done using any programming language such as Java programming or python programming.
3. For SHA256 hashing, you may use library compatible as per your programming language.

THEORY:

In cryptography, ensuring data integrity and confidentiality is crucial. SHA-256 is a cryptographic hash function that takes an input of any size and produces a fixed 256-bit output. It is a one-way function, meaning it cannot be reversed, making it ideal for verifying data integrity, ensuring that the message hasn't been altered during transmission.

AES (Advanced Encryption Standard) is a symmetric encryption algorithm used to secure data by encrypting it with a secret key. AES uses the same key for both encryption and decryption and supports key sizes of 128, 192, or 256 bits. It's widely used for securing communication, data storage, and more.

In this practical, SHA-256 is used to generate a hash of the original message to check its integrity, while AES encrypts the message and the hash for confidentiality. The same secret key is used to decrypt the bundle and verify the message integrity, ensuring secure communication.

CODE:

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.util.Base64;

public class trialfile {
    public static void main(String[] args) throws Exception {
        //AES key (128-bit)
        KeyGenerator keyGen = KeyGenerator.getInstance("AES");
        keyGen.init(128);
        SecretKey secretKey = keyGen.generateKey();

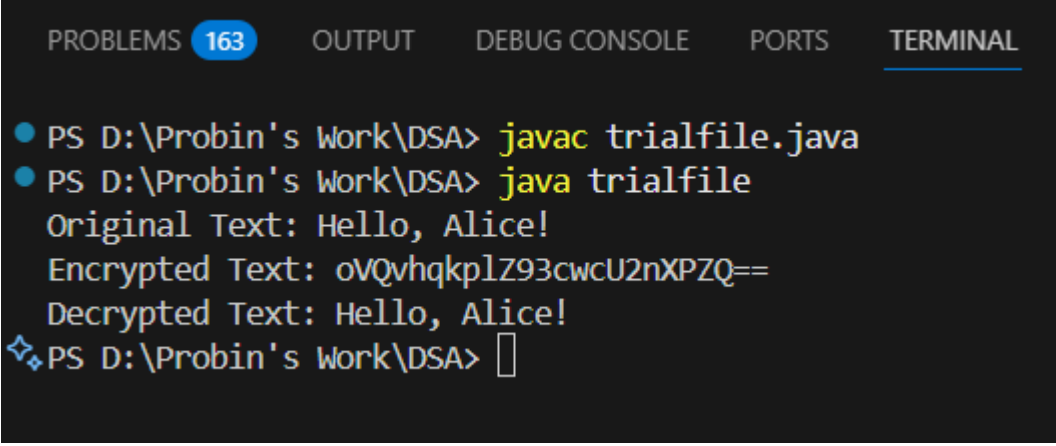
        // Plaintext
        String plaintext = "Hello, Alice!";
        System.out.println("Original Text: " + plaintext);

        // Encryption
        String encryptedText = encrypt(plaintext, secretKey);
        System.out.println("Encrypted Text: " + encryptedText);

        // Decryption
        String decryptedText = decrypt(encryptedText, secretKey);
        System.out.println("Decrypted Text: " + decryptedText);
    }

    // AES Encryption
    public static String encrypt(String plaintext, SecretKey secretKey) throws Exception {
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedBytes = cipher.doFinal(plaintext.getBytes());
        return Base64.getEncoder().encodeToString(encryptedBytes);
    }

    // AES Decryption
    public static String decrypt(String encryptedText, SecretKey secretKey) throws Exception {
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(encryptedText));
        return new String(decryptedBytes);
    }
}
```

OUTPUT:

```
PROBLEMS 163 OUTPUT DEBUG CONSOLE PORTS TERMINAL
• PS D:\Probin's Work\DSA> javac trialfile.java
• PS D:\Probin's Work\DSA> java trialfile
Original Text: Hello, Alice!
Encrypted Text: oVQvhqkplZ93cwcU2nXPZQ==
Decrypted Text: Hello, Alice!
❖ PS D:\Probin's Work\DSA> 
```

LATEST APPLICATIONS:

1. **Secure Web Browsing (HTTPS):**
SHA-256 is commonly used in SSL/TLS protocols, where it plays a critical role in creating secure connections between web browsers and servers. It is used in the creation of digital certificates, ensuring the integrity and authenticity of websites. This allows users to trust that the websites they are visiting have not been tampered with.
2. **Digital Signatures and Certificates:**
SHA-256 is widely used in digital signatures for verifying the integrity of messages and documents. It ensures that data has not been altered during transmission. Popular services, including email authentication, software updates, and legal documents, rely on SHA-256 for generating unique signatures that represent the data.
3. **Cryptocurrency and Blockchain Technology:**
SHA-256 is fundamental to the functioning of blockchain and cryptocurrency, particularly Bitcoin. It is used in the process of mining and generating blocks, ensuring that transactions are secure and that the blockchain remains tamper-proof. SHA-256's one-way hashing function ensures that data is securely stored and verified within the blockchain.
4. **Password Storage and Authentication:**
In modern systems, SHA-256 is used to securely hash passwords before they are stored in databases. It ensures that even if a database is compromised, the actual passwords remain safe, as only the hashed values are stored. Many secure authentication mechanisms use SHA-256 hashing for creating strong password verification processes.

LEARNING OUTCOME:

1. **Understand SHA-256 Algorithm:** Learn how SHA-256 generates a fixed-size hash from variable input and its role in ensuring data integrity.
2. **Apply AES Encryption:** Implement AES encryption and decryption with a secret key to secure data, while using SHA-256 to verify message integrity.
3. **Develop Secure Communication Systems:** Gain experience in applying encryption (AES) and hashing (SHA-256) to create secure communication systems.
4. **Understand Key Management in Cryptography:** Recognize the importance of key management in symmetric encryption for maintaining security.
5. **Enhance Cryptography Skills:** Develop practical skills in cryptography, including hashing and encryption, to secure data effectively.

REFERENCES:

1. <https://www.geeksforgeeks.org/sha-256-in-cryptography/>
2. https://www.tutorialspoint.com/cryptography/aes_encryption.htm
3. <https://www.geeksforgeeks.org/sha-256-hash-in-java/>
4. <https://www.openssl.org/docs/manmaster/man7/AES.html>